

الجمهورية الجزائرية الديمقراطية الشعبية

ALGERIAN DEMOCRATIC AND POPULAR REPUBLIC

وزارة التعليم العالي والبحث العلمي

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

جامعة غرداية

Registration N°

University of Ghardaia

/...../...../...../...../.....



كلية العلوم والتكنولوجيا

Faculty of Science and Technology

قسم الرياضيات والإعلام الآلي

Department of Mathematics and Computer Science

End of study thesis, with a view to obtaining the

Master's degree

Domain: Mathematics and Computer Science

Sector: Computer Science

Specialty: Intelligent Systems For Knowledge Extraction

Theme

Few-Shot Learning: Learning with less data

Presented by:

BENAHMED Zakaria

BELLAOUAR Mahmoud

Publicly supported on: 18 / 06 / 2022

Examined by the jury composed of:

Mr. Abderrahman Adjila	MAA	Univ. Ghardaia	PRESIDENT
Mr. Youcef Mahjoub	MAA	Univ. Ghardaia	SUPERVISOR
Mr. Houssam Eddine Degha	MAB	Univ. Ghardaia	EXAMINER

ACADEMIC YEAR: 2021/2022

Dedication

“

*Praise be to **Allah**, who has the favour and who helped me and gave me strength to reach this moment,*

*Dedicated to my late **mother** the real gem, may **Allah** be merciful to her. She was always helping me with all she could, guided me to stay away from problems and encouraged me to complete my studies at all stages of her life. **Mom**, this is for you,*

*To my **father**, who did not hesitate to provide everything I needed, taught me many things, and supported me at all times. For me, thank you is not enough for everything you have given me,*

*To my **brothers** and **sisters**, thanks for all those laughs, and for all that you did for me,*

*To all my **friends** who have always been there for me, even in the tough days, please accept my deepest gratitude...*

Thank you to all.

”

- **Mahmoud**-

“

*First and foremost, thanks always to **Allah**, who enabled me
to accomplish this work,*

*I dedicate this work especially to my **grandmother**, may allah
have mercy and forgiveness to her,*

*To my **mother** and **father** who were the main support for me
to do this work,*

*To my **brothers** and **sisters** who have supported me on this
journey,*

*To all my distinguished **teachers** who taught me a lot from
elementary school to university..*

”

- **Zakaria**

Acknowledgement

First and foremost, thanks always to Allah, who enabled us to accomplish this work. I would like to thank my teacher and supervisor **Mr. Youcef Mahdjoub** for all the support and encouragement he provided and to continue working with us until the last moment and sacrifice in order to accomplish this thesis. Thank you for everything

Zakaria—

First and foremost, I thank Allah for giving me the hope that led me to go on.

I am indebted to my teacher and supervisor **Mr. Youcef Mahdjoub** for his patience through this process, unwavering support, and encouragements . I hope you find some kind of satisfaction in this modest work. Thank you so much.

Mahmoud—

This is acknowledgement to all the teachers who helped us and were patient in teaching and supporting us and they gave everything to guide us and ensure the development of our abilities and skills.

Thanks to all the teachers for sharing their expertise with us, and to the professors, members of the jury, for accepting the discussion and evaluation of this modest thesis, and we hope that you will find some satisfaction in this thesis. We're so grateful.

Zakaria and Mahmoud—

Abstract

Deep learning has achieved great success in solving many tasks in various fields, but this success is conditional on the availability of a significant amount of data to train on, as well as the availability of the computing capabilities necessary to obtain good results, while in reality, there are several problems and areas of nature that have a small amount of data, and here comes the role of our field of study, Few-Shot learning, which attempts to bridge the gap between deep learning models and human learning ability from a few examples. In our work, we try to discover the field of Few-Shot learning, first theoretically, by taking note of its most important foundations, approaches and methods. Secondly, practically, by training the Siamese neural network with a small amount of data, so that our model is able to distinguish classes that were not seen during training, we recorded an important observation, that good discrimination is not coupled with extensive training and that satisfactory results regarding discrimination accuracy reach about 84%.

Keywords : Few-Shot Learning, Meta learning, Metric Learning, Transfer learning, Deep learning, Siamese neural network.

Résumé

L'apprentissage profond a obtenu un grand succès dans la résolution de nombreuses tâches dans divers domaines, mais ce succès est conditionné par la disponibilité d'une quantité importante de données sur lesquelles l'entraînement est effectué, ainsi que par la disponibilité des capacités de calcul nécessaires pour obtenir de bons résultats, alors qu'en réalité, plusieurs problèmes et domaines de la nature ne disposent que d'une petite quantité de données, et c'est là qu'intervient le rôle de notre domaine d'étude, l'apprentissage avec peu d'exemples, qui tente de faire le rapprochement entre les modèles d'apprentissage profond et la capacité d'apprentissage humaine à partir de quelques exemples. Dans notre travail, nous essayons de découvrir le domaine de l'apprentissage avec peu d'exemples, d'abord théoriquement, en prenant note de ses fondements, ses approches et ses méthodes les plus importantes. Deuxièmement, de manière pratique, en entraînant le réseau de neurones siamois avec une petite quantité de données, de sorte que notre modèle soit capable de distinguer des classes qui n'ont pas été vues pendant l'entraînement. Nous avons enregistré une observation importante, à savoir qu'une bonne discrimination n'est pas associée à un entraînement intensif et que les résultats satisfaisants concernant l'exactitude de la discrimination atteignent environ 84%.

Mots clés : Apprentissage avec peu d'exemples, Méta apprentissage, Apprentissage métriques, Apprentissage par transfert, Apprentissage profond, Réseau de neurones siamois.

ملخص

التعلم العميق حقق نجاحا كبيرا في توفير الحلول للعديد من المهام في مختلف المجالات ولكن هذا النجاح مشروط بتوفر كمية معتبرة من البيانات للتدريب عليها وكذلك توفر قدرات الحوسبة اللازمة للحصول على نتائج جيدة، بينما واقعا هناك عدة مشكلات ومجالات بطبيعتها تتوفر على كمية قليلة من البيانات وهنا يأتي دور مجال دراستنا التعلم من خلال عدد قليل من البيانات، والذي يحاول سد الفجوة بين نماذج التعلم العميق وقدرة التعلم البشري من أمثلة قليلة. في هذا العمل حاولنا اكتشاف مجال التعلم من خلال عدد قليل من البيانات نظريا من خلال الاحاطة بأهم أسسه ومقارباته وطرائقه ومن الناحية التطبيقية من خلال تدريب الشبكة العصبية السيامية باستعمال عدد قليل من البيانات بحيث يكون نموذجنا قادراً على التمييز بين أصناف لم يرها خلال التدريب وقد سجلنا ملاحظة مهمة أن التمييز الجيد غير مقرون بالتدريب المكثف ونتائج مرضية بخصوص دقة التمييز وصلت لنحو 84 بالمئة.

كلمات مفتاحية : التعلم من خلال بيانات قليلة، ميتا التعلم، التعلم المتري، التعلم المتنقل، التعلم العميق، الشبكة العصبية السيامية.

Contents

Dedication	I
Acknowledgement	III
Abstract	IV
Résumé	V
VI	ملخص
Introduction	1
1 Few-Shot Learning Preliminary Concepts	2
1.1 Deep learning	3
1.2 What is Few-Shot-Learning ?	3
1.3 Few-Shot Learning Variations	4
1.4 Few-Shot Learning challenges	4
1.5 Few-Shot learning vs standard supervised learning	4
1.6 Meta-learning	5
1.6.1 Optimization-based Meta-Learning	6
1.6.2 Metric-based Meta-Learning	7
1.7 Pre-training and Transfer learning	8
1.8 Fine-tuning for Few-Shot learning	9
1.9 Datasets used for Few-Shot learning	10
1.9.1 Omniglot	10
1.9.2 Mini-ImageNet	10
1.9.3 Fewshot-CIFAR	11
1.9.4 CUB-200-2011	11
1.10 Conclusion	11
2 Related Work	12
2.1 Introduction	13
2.2 Data augmentation Techniques	13
2.2.1 Squared gradient magnitude loss (SGM)	13
2.2.2 Image Deformation Meta-Networks (IDeMe-Net)	14
2.3 Gradient-based Meta-learning Techniques	15
2.3.1 Meta-Learning LSTM (Long short-term memory)	15
2.3.2 Model-Agnostic Meta-Learning (MAML)	16
2.3.3 Model-Agnostic Meta-Learning ++ (MAML++)	17
2.3.4 Reptile	17
2.3.5 Meta-SGD	17

2.4	Metric-Based Techniques	18
2.4.1	Siamese Neural Networks	18
2.4.2	Matching Networks	19
2.4.3	Prototypical Networks	20
2.4.4	Relation Network	21
2.5	Application	22
2.5.1	Computer Vision	22
2.5.2	Natural Language Processing	22
2.5.3	Robotics	22
2.5.4	Speech Recognition	23
2.6	Conclusion	23
3	Experiment	24
3.1	Introduction	25
3.2	Experimentation roadmap	25
3.3	Dataset	26
3.4	Environment	27
3.4.1	Software	27
3.4.2	Hardware	27
3.5	Architecture	28
3.6	Results and Discussion	29
3.6.1	Preprocessing	29
3.6.2	Experimental results	30
3.6.3	Visualization	32
3.7	Conclusion	34
	Conclusion	35

List of Figures

1.1	Few-Shot Image Classification [31].	3
1.2	General framework of meta learning [19].	5
1.3	Example of Metric-based technique [16].	8
1.4	Training stage [4].	9
1.5	Fine-tuning stage [4].	9
1.6	Omniglot Dataset.	10
2.1	SGM loss [10].	13
2.2	Architecture of image deformation meta-network (IDeMe-Net)[6].	14
2.3	LSTM meta-learner model training [23].	15
2.4	Algorithm Of Model Agnostic Meta Learning (MAML), θ represent quickly adapt to new tasks [13].	16
2.5	Process of Meta-SGD, meta space (θ, α) that learns the meta-learner accelerated learning is performed by the meta-learner in a learner space θ in which learners learn with a specific task [18].	18
2.6	A simple Siamese Neural Networks for binary classification [15].	19
2.7	Matching Networks architecture [26].	20
2.8	Prototypical Networks [24].	21
2.9	Relation Network [25].	22
3.1	MINST-FASHION Data set [29].	26
3.2	Our Siamese network architecture.	28
3.3	Feature vector CNN's Encoder.	28
3.4	The training and testing accuracy and loss plots using 10 batch.	29
3.5	The graph for the performance over 20 epochs.	31
3.6	The graph for the performance over 50 epochs.	31
3.7	The graph for the performance over 100 epochs.	31
3.8	The confusion matrix for the settings of 10 batch size and 20 epochs performance on the train set.	32
3.9	The confusion matrix for the settings of 10 batch size and 20 epochs performance on the test set.	32
3.10	The confusion matrix for the settings of 10 batch size and 20 epochs performance on the evaluation set.	33
3.11	Visualization of differentiate between classes from evaluation set.	33

List of Tables

1.1	Equation Terms.	5
2.1	Accuracy Of SGM Model.	14
2.2	Accuracy Of IDeMe-Net Model.	14
2.3	Accuracy Of Meta-Learning LSTM Model.	16
2.4	Accuracy Of Model-Agnostic Meta-Learning (MAML).	17
2.5	Accuracy Of Model-Agnostic Meta-Learning++ (MAML++).	17
2.6	Accuracy Of Reptile.	17
2.7	Accuracy Of Meta-SGD.	18
2.8	Accuracy Of Siamese Neural Networks.	19
2.9	Accuracy Of Matching Network.	20
2.10	Accuracy Of Prototypical Networks.	21
2.11	Accuracy Of Relation Network.	22
3.1	Google Colab used requirements.	27
3.2	The accuracy of the training and testing and evaluation set.	30

List of Abbreviations

FSL	<i>Few-Shot Learning.</i>
OSL	<i>One-Shot Learning.</i>
ZSL	<i>Zero-Shot Learning.</i>
SGM	<i>Squared Gradient Magnitude Loss.</i>
IDeMe-Net	<i>Image Deformation Meta-Networks.</i>
MAML	<i>Model-Agnostic Meta-Learning.</i>
MAML++	<i>Model-Agnostic Meta-Learning++.</i>
CNN	<i>Convolutional Neural Network.</i>
LSTM	<i>Long Short-Term Memory.</i>
K-NN	<i>K-Nearest Neighbors Algorithm.</i>
NLP	<i>Natural Language Processing.</i>

Introduction

Contents

Modern artificial intelligence has made significant progress in many domains, thanks to advanced deep learning algorithms, the computing power and availability of large data sets.

As a result of these advances, deep learning models are performing very well in various fields such as image classification, video games, self-driving cars, natural language processing and other domains. However, a large quantity of labeled dataset is needed in order to realize these achievements. On the other hand, humans have a great capacity to learn from a small number of examples, and not only that, but they also have the ability to generalize this learning to new problems through what they have previously learned, for example, a child can recognize an object that he has only seen once or twice. This is what deep learning models can't do.

In order to bridge the gap between human learning ability and deep learning models, Few-shot learning (FSL) has been introduced to accomplish this, to solve problems when little or no data is available or when we lack the computational capabilities to learn, in general to deal with data scarcity in any field.

Researchers have worked hard on Few-shot learning, which has led to the emergence of many approaches and techniques, such as: Data augmentation techniques, metric-based techniques, gradient-based meta-learning techniques and more. Each of these techniques has a specific way of solving learning problems from few data.

In this work, we explore the study related to Few-shot learning and the most important ways in which data scarcity was addressed, we also study the generalization through a few examples using the Siamese neural network.

This document is divided into three chapters as follows:

- Chapter One: We start by talking about deep learning, then we define Few-shot learning and its variations, the different methods, some basic mathematics used in these methods and finally the datasets used in Few-shot learning.
- Chapter Two: It is about related work in Few-shot learning, defining the techniques used, their results and we also mentioned important applications in the field.
- Chapter Three: The experimental part of our work is based on one of the previously mentioned methods, which is the Siamese neural network, using a little bit of training data from the Fashion MNIST dataset to study the effect of generalization on classes that we have not trained on, with different settings such as the number of epochs and the batch size.

Few-Shot Learning Preliminary Concepts

Contents

1.1	Deep learning	3
1.2	What is Few-Shot-Learning ?	3
1.3	Few-Shot Learning Variations	4
1.4	Few-Shot Learning challenges	4
1.5	Few-Shot learning vs standard supervised learning	4
1.6	Meta-learning	5
1.6.1	Optimization-based Meta-Learning	6
1.6.2	Metric-based Meta-Learning	7
1.7	Pre-training and Transfer learning	8
1.8	Fine-tuning for Few-Shot learning	9
1.9	Datasets used for Few-Shot learning	10
1.9.1	Omniglot	10
1.9.2	Mini-ImageNet	10
1.9.3	Fewshot-CIFAR	11
1.9.4	CUB-200-2011	11
1.10	Conclusion	11

1.1 Deep learning

Deep learning is a branch of machine learning characterized by the presence of a deep neural network, which can be described as a model with several hidden layers. In order to make decisions, this model must be trained on a large amount of data, unlike the human brain which can learn and perceive from only a few examples. Few-shot learning (FSL) is a concept that consists in building algorithms that can bring this human ability of learning from just a small amount of data, this is what this work is about.

1.2 What is Few-Shot-Learning ?

FSL generally focuses on the N-way K-shot classification tasks, that is to train the model with a few labeled examples (K) for N new classes that requires fast model adaptation to new tasks.

Few-shot learning can be categorized into almost four approaches: generative models, feature or metric learning, data augmentation, and meta-learning [30, 31].

Some references have reduced the number to only three, as follows: Data, Model, Algorithm [28].

Let's define an N-way-K-Shot-classification:

1. a support set or training set that composed of :
 - (1) N class labels.
 - (2) K labeled few images of each class.
2. a query set composed of Q query images. (see Figure (1.1))



Figure 1.1: Few-Shot Image Classification [31].

1.3 Few-Shot Learning Variations

In general, researchers are working on three different variation in **FSL**:

1. Few-Shot Learning (FSL).
2. One-Shot Learning (OSL).
3. Zero-Shot Learning (ZSL).

The only difference between different FSL variations is the value of N , in the case of one-shot N becomes one and zero for zero-shot learning, where the number of samples to train with is N , and the number of classes to train on is K .

1.4 Few-Shot Learning challenges

FSL works to solve problems in effective ways, and this is what makes it have advantages, which are:

- **Learning when data is scarce:** When we face the problem of data scarcity and cannot get enough examples, the best solution is to resort to the use of FSL, which has the ability to address these situations, for example trying to know and discover a new drug and find out if its components that it contains are toxic or non-toxic.
- **Reduce data and processing time:** The approach that FSL depends on helps reduce the size of data in the future, including processing time, especially for images.
It will have an important role in making models more efficient and effective.
- **Simulation of the human brain:** Learning without the need for much information is what distinguishes the human mind in this sense.

The same thing was applied to the machine to be closer in the way the human mind works.

1.5 Few-Shot learning vs standard supervised learning

In this section we mention some of the differences between Supervised learning and FSL.

1. Supervised learning:

- In order to predict new samples, required a lot of experience.
- All data in a dataset has a label.
- Single task learning.

2. Few-shot learning:

- Attempts to predict new samples without much experience.
- Not all data in dataset have a label.
- Meta learning's goal is learning to learn, learn by himself, learning from a few experiences.

1.6 Meta-learning

We now begin to explain about **Meta-learning**, which has shown a lot of progress in FSL, it is also worth mentioning that there are approaches in FSL that are not based on **Meta-learning**.

Meta-Learning: or we can call it here (learn to learn) the idea of using this is to acquire knowledge or inductive biases [5] also learn to optimize deep models [19].

Also it learns from tasks and then adapts to new tasks T as it is in the Figure (1.2), it is further used to deal with FSL problems and it is taken as prior knowledge in order to guide each specific task of the FSL [28].

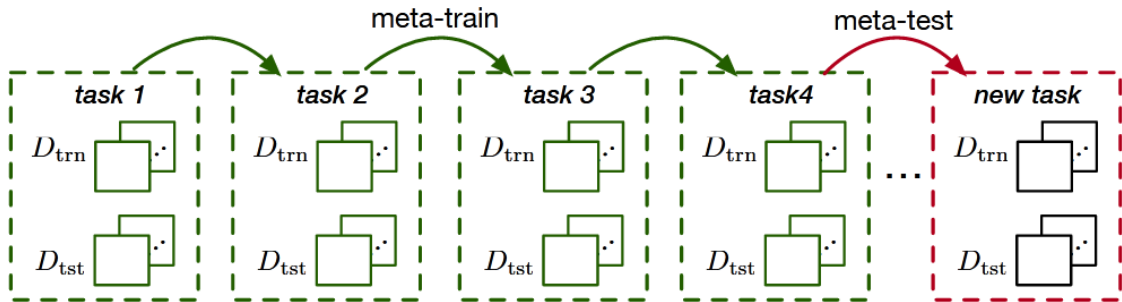


Figure 1.2: General framework of meta learning [19].

Meta-Learning can also be relied upon for several reasons, such as:

- Adapt and learn from new tasks quickly.
- Ability to build generalizable systems.

In the following table 1.1, we put some symbols and their appropriate description in order to facilitate understanding of the equations.

Table 1.1: Equation Terms.

<i>Notation</i>	<i>Description</i>
\mathcal{D}^{train}	<i>Training set</i>
\mathcal{D}_i^{train}	<i>Training set for task T_i</i>
\mathcal{D}^{test}	<i>Test set</i>
\mathcal{D}_i^{test}	<i>Test set for task T_i</i>
\mathcal{T}_i	<i>Task i</i>
\mathcal{L}	<i>loss function</i>
f_θ	<i>Function (model) with parameters θ</i>
g_ϕ	<i>Meta-Learning model with parameters ϕ</i>
α	<i>Learning rate</i>
θ	<i>Parameters</i>
θ^*	<i>Optimal parameters</i>

Meta-learning is based on the approximation of the f function with the parameters θ as in the following equation (1.1):

$$y \approx f(\mathcal{D}_i^{\text{train}}, x; \theta), \text{ where } (x, y) \in \mathcal{D}_i^{\text{test}} \quad (1.1)$$

so its goal is to make the performance of any task t_i randomly with a perfect distribution of tasks $p(T)$.

1.6.1 Optimization-based Meta-Learning

This method explains how the training data, which is limited, can be improved while ensuring an acceptable generalization performance, by taking advantage of the meta-learning structure and its algorithms, all of which allows improvement on the work of limited training examples [22].

Optimization-based approaches go through two phases:

- **Learner:** The learner model f_θ is a specific task and trained on a specific task, but this few-shot task does not generalize if the learned model trains from scratch using gradient descent.
- **Meta-Learner:** Works during task assignment training $T \approx p(T)$ as the meta-learner learns (ϕ) to update the parameters of the learner model (θ) through the training set $\mathcal{D}^{\text{train}}$.

$$\theta^* = g_\phi(\theta, \mathcal{D}^{\text{train}}) \quad (1.2)$$

The meta-learner tries to produce the updated learner model parameters θ^* provided they are better than θ .

The process of optimization during the training of the meta learner improves ϕ for the meta learner and θ for the individual training tasks and upon completion prior knowledge is included ϕ and then occurs θ for the test task.

- a) **LSTM Meta-Learner** Section(2.3.1): Use the gradient descent of the neural network f to update its parameters θ and the goal of all this is to try to train $f(\theta)$ on the data $\mathcal{D}^{\text{train}}$. (1.3)

$$\theta_{i+1} = \theta_i - \alpha \nabla f(\theta_i) \quad (1.3)$$

Work with the function $g(\phi)$ that acts as an enhancer instead of **SGD** and we do not have to fix the learning rate α .

$$\theta_{i+1} = \theta_i + g_i(\nabla f(\theta_i); \phi) \quad (1.4)$$

The equation (1.4) shows not using **SGD** and therefore to dispense with α and use $g(\phi)$. The $g(\phi)$ meta learner is also modeled as an LSTM in order to suggest parameters for the f learner Figure(2.3).

$$\theta_{i+1} = g_i(\nabla f(\theta_i), \theta_i; \phi) \quad (1.5)$$

b) **Model-Agnostic Meta-Learning (MAML)** Section(2.3.2): It aims to have good θ parameters and this makes the optimization faster in the next steps from the computed gradient descent and from small data, through the assignment of tasks $p(T)$ these parameters θ are learned.

Compared with the **LSTM meta-learner**, **MAML** has one model with parameters θ . It also seeks to reach the optimal parameters θ^* during the proposed task.

We have the function f with the parameter θ when adapting a new task T_i becomes θ'_i . In calculating in order to update the parameters θ'_i equation (1.6) we use more than once gradient descent updates on task T_i (sometimes once is enough)[9].

$$\theta'_i = \theta - \alpha \nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_{\theta}) \quad (1.6)$$

The model parameters are trained by optimizing for the performance of $f_{\theta'_i}$ with respect to θ across tasks sampled from $p(T)$, the meta-objective is in following equation (1.7):

$$\min_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta - \alpha \nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_{\theta})}) \quad (1.7)$$

The loss function \mathcal{L} helps to learn from the training data and to measure how well the model works. Therefore, in the above-mentioned equation (1.7), the **min** function was proposed in order to reduce the value of the loss function \mathcal{L} with respect to θ across tasks sampled from.

1.6.2 Metric-based Meta-Learning

Metric learning its purpose is to determine the similarity or dissimilarity between data samples based on a distance metric as Euclidean distance (1.8):

$$\sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (1.8)$$

The properties of a distance metric are :

(a) Non-negativity: $d(x, y) \geq 0$,

- (b) Identity of Non-discernibles: $d(x, y) = 0$ if and only if $x = y$,
- (c) Symmetry: $d(x, y) = d(y, x)$, and
- (d) Triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$

Metric learning is the key idea behind nearest neighbors algorithms (k-NN), Therefore before the calculation process we use an embedding function f to find the embedding vectors.

in the case of Few-shot learning, metric-based techniques combine good characteristics between parametric models and nonparametric with the contribution of Meta-Learning to fast adaptability of new tasks.

These methods are divided according to this reference [17] into three groups :

- learning class representations represented by Prototypical Network.
- learning distance by Relation Network.
- learning feature embeddings for Siamese Network and Matching Network.

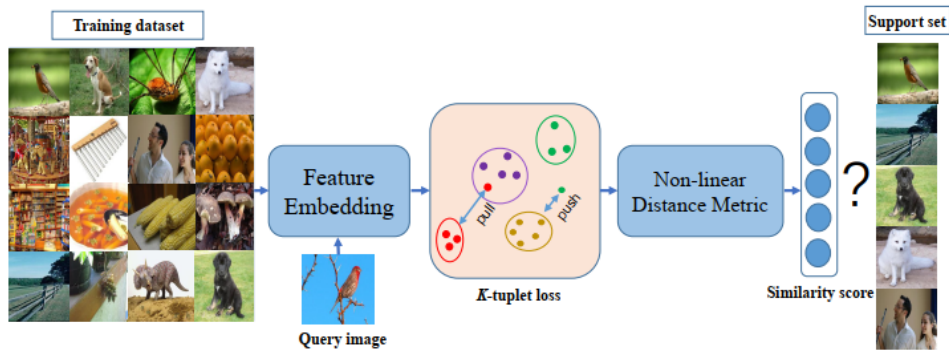


Figure 1.3: Example of Metric-based technique [16].

1.7 Pre-training and Transfer learning

Transfer learning takes advantage of previous models to get more efficient, it adapts to training new models related to the previous model, and it works well with small data and improves with **pre-training**.

In the field of **pre-training** images, studies have shown that an increase in the size of data has the benefits of **transfers** in performance [7].

The Figure(1.4) shows the training stage, starting from a base class data and ending with the predictive function [4].

f_θ is a feature extractor.

$C(\cdot | \mathbf{W}_b)$ is a classifier, with parameters θ , which is a network parameter.

Whereas the classifier is parameterized by matrix weights ($\mathbf{W}_b \in \mathbb{R}^{d \times c}$).

Using $\mathbf{x}_i \in \mathbf{X}_b$ the classifier works to minimizing cross-entropy classification loss from the beginning using training examples from the base classes.

We refer to the letter \mathbf{d} as the dimension of the encoded feature, and the letter \mathbf{c} is the number of output classes.

The classifier consists of a linear layer $\mathbf{W}_b^\top f_\theta(\mathbf{x}_i)$ in addition to the SOFTMAX¹ function.

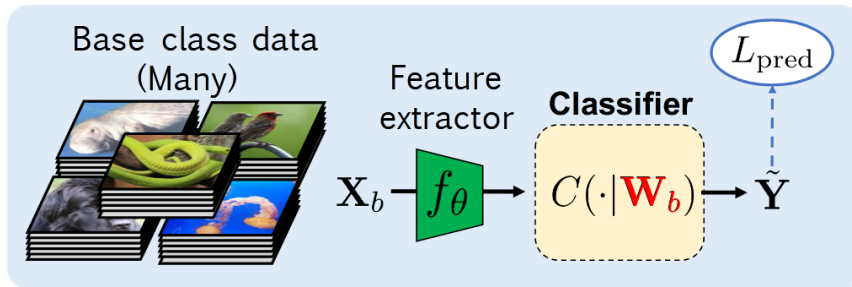


Figure 1.4: Training stage [4].

Extensive **pre-training** is done on the large dataset and then **fine-tuned** to the target data [3], **pre-training** and **fine-tuning** had similar steps.

1.8 Fine-tuning for Few-Shot learning

A strong foundation for **FSL** is the fine tuning especially used on the cross-entropy loss function equation (1.9), because is a strong baseline for **FSL** [8].

$$\hat{\theta} = \arg \min_{\theta} \frac{1}{N_m} \sum_{(x,y) \in \mathcal{D}_m} -\log p_{\theta}(y | x) + R(\theta) \quad (1.9)$$

The model follows the standard transfer learning procedure for **fine-tuning** and network **pre-training**.

During the **fine-tuning** phase to configure the model in order to identify new classes, f_θ which is the feature extractor, we fix the network parameters θ and train a new classifier $C(\cdot | \mathbf{W}_b)$.

And by using a few examples labeled in new classes \mathbf{X}_n Figure(1.5) we reduce the output function [4].

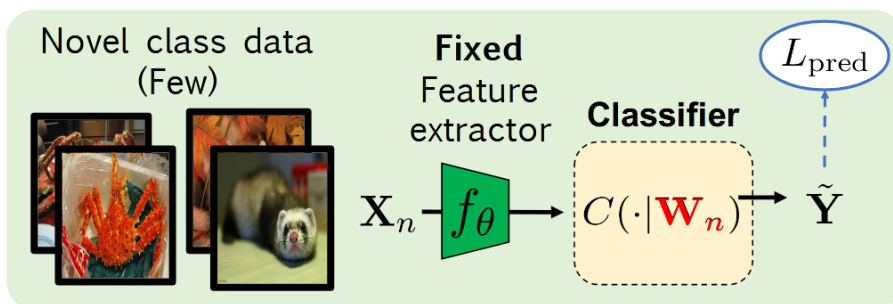


Figure 1.5: Fine-tuning stage [4].

¹(<https://proceedings.neurips.cc/paper/1989/file/0336dcbab05b9d5ad24f4333c7658a0e-Paper.pdf>)

1.9 Datasets used for Few-Shot learning

In this section, we explain the different types of datasets used in the field, and we start with:

1.9.1 Omniglot

Omniglot² is one of the most widely used dataset and it is small. It is a handwritten dataset that is very similar to **MNIST Handwritten**³. **MNIST** has 10 classes, each class has 6000 samples, but **Omniglot** has 1000 classes, each class has only 20 samples as in the Figure (1.6). This makes classification difficult in **Omniglot** compared to **MNIST**.

Omniglot contains alphabets for 50 languages, and this makes the total of letters for all languages in this dataset is 1623 letters, and each letter in a specific language is repeated 20 times in different handwritings, and the sample of one letter is an image with a scale of 105 X 105 pixels.

A training set is an alphabet of 30 languages, and this makes the number of training letters 964 (class). The evaluation set contains the alphabet for 20 languages, with a total number of letters up to 659 characters are the classes/labels [12].

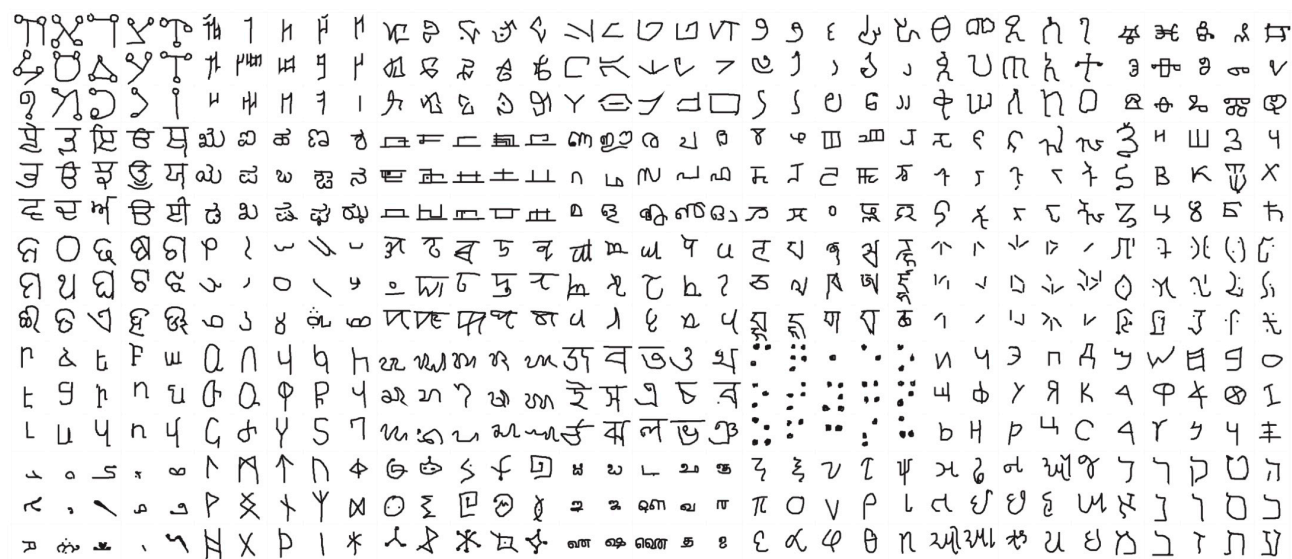


Figure 1.6: Omniglot Dataset.

1.9.2 Mini-ImageNet

Mini-ImageNet⁴ contains 100 classes, each class contains 600 samples with a total of 60 thousand colored images of size 84×84 , and it is a small version of **ImageNet**.

- a) **ImageNet** is the most famous and largest data set for classifying images. It has more than 20 thousand classes with a total of more than 14 million colored images.

²(<https://github.com/brendenlake/omniglot>)

<https://www.tensorflow.org/datasets/catalog/omniglot>)

³(<https://www.tensorflow.org/datasets/catalog/mnist>)

⁴(<https://github.com/yaoyao-liu/mini-imagenet-tools#about-mini-ImageNet>)

- b) **Mini-ImageNet** was proposed in order to reduce the size and allow models to be trained for experiments [12].

1.9.3 Fewshot-CIFAR

Fewshot-CIFAR or **FC-CIFAR** it is a dataset based on CIFAR⁵, and it is like a **Mini-ImageNet** that has been reduced for experiments.

It is also considered faster than a **Mini-ImageNet** because of the reduction in the size of the image, this dataset contains 60 thousand color images, size 32 x 32 per image, it differs in the division of classes according to the type of dataset **Fewshot-CIFAR10** contains 10 classes and **Fewshot-CIFAR100** contains 100 classes [21].

1.9.4 CUB-200-2011

This dataset is specific to the characteristics of birds and includes more than 11 thousand images, and these images are colored for 200 species of birds [27], each image contains labeled attributes [12].

We have mentioned these types of datasets because they are the most famous in this field, although there are many datasets that we did not mention, and all of them have their own advantages and uses.

1.10 Conclusion

During this chapter, we introduced Few-Shot learning (FSL), variations of FSL like Zero-shot and One-shot, and explained the difference between FSL vs standard supervised learning.

Then we introduced Meta Learning and put some mathematical concepts that it works with, and we explaining the Metric-based Meta-Learning and the definition of distance metric.

We also defined Pre-training and Transfer learning and their work. We added the definition of Fine-tuning for FSL, and finally we defined the most famous FSL dataset.

In the next chapter, we explain the methods adopted in each approach, the way they work, and the results achieved.

⁵(<https://www.cs.toronto.edu/~kriz/cifar.html>)

Related Work

Contents

2.1	Introduction	13
2.2	Data augmentation Techniques	13
2.2.1	Squared gradient magnitude loss (SGM)	13
2.2.2	Image Deformation Meta-Networks (IDeMe-Net)	14
2.3	Gradient-based Meta-learning Techniques	15
2.3.1	Meta-Learning LSTM (Long short-term memory)	15
2.3.2	Model-Agnostic Meta-Learning (MAML)	16
2.3.3	Model-Agnostic Meta-Learning ++ (MAML++)	17
2.3.4	Reptile	17
2.3.5	Meta-SGD	17
2.4	Metric-Based Techniques	18
2.4.1	Siamese Neural Networks	18
2.4.2	Matching Networks	19
2.4.3	Prototypical Networks	20
2.4.4	Relation Network	21
2.5	Application	22
2.5.1	Computer Vision	22
2.5.2	Natural Language Processing	22
2.5.3	Robotics	22
2.5.4	Speech Recognition	23
2.6	Conclusion	23

2.1 Introduction

In FSL, there are many approaches and techniques, we discuss the most popular ones as: **data** or **data augmentation** techniques, **gradient-based meta-learning** techniques and **metric-based** techniques, which will be explained below:

2.2 Data augmentation Techniques

Before starting to explain what data augmentation is, we start with a definition of **augmentation**: it is a way to increase the number of training samples, for example when we talk about the compute vision domain the basic of augmentation has several operations, including: translation, adding noise into images, rotating, flipping, cropping [19], zooming and changing the brightness level.

These are some suggested augmentation that work on custom networks and models to generate more samples from the few existing ones.

The aim of all this is to improve the quality of dataset, especially for new uses in practical life in the medical field. **Data augmentation** can improve the performance of models and expand limited datasets to take advantage of the capabilities of big data.

Although increasing data is a good thing to solve few data problems, but it also has some limitations, for example, if your current data distribution is unequal, it will increase the data distribution to be unequal as well, there is also a high chance of over-fitting as the **data augmentation**.

2.2.1 Squared gradient magnitude loss (SGM)

This method generates new data for data augmentation for a few shot classes with a new loss function. The goal of SGM loss is to have a good feature extractor so as to enable effective classifiers to learn from a few examples. It is also one of its goals to reduce the difference between trained classifiers [10].

As the Figure (2.1) shows the classifier \mathbf{W} in good and bad case.

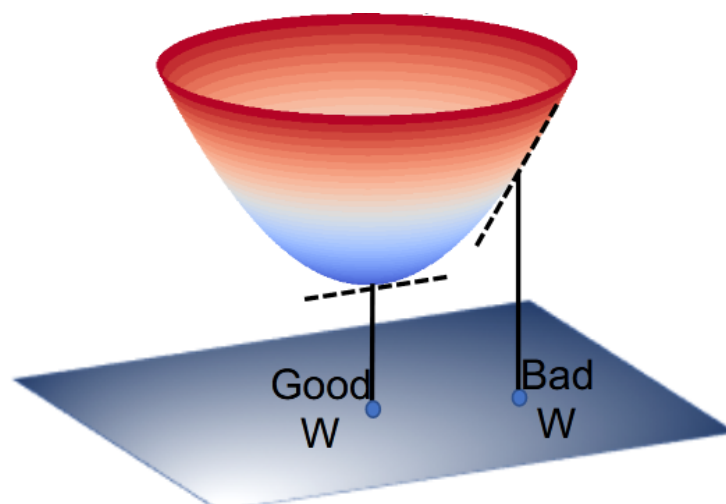


Figure 2.1: SGM loss [10].

The table 2.1 shows the result of **SGM** model in Mini-ImageNet (1.9.2) dataset.

Table 2.1: Accuracy Of SGM Model.

MiniImageNet Dataset 5- Way Accuracy (%)				
Model	Technique	1-shot	5-shots	10-shots
SGM	Data Augmentation	45.1	72.7	79.1

2.2.2 Image Deformation Meta-Networks (IDeMe-Net)

This method works well on One-shot learning, but it also works with **FSL**, and it depends on two modules [6]:

- **Embedding Sub-network.**
- **Deformation Sub-network.**

Embedding Sub-network: It comprises of a deep convolutional network for the purpose of feature extraction and a non parametric one-shot classifier, the input is image use a residual network [11] to produce its feature representation.

Deformation Sub-network: It explores interaction and integration between the gallery images and the probe images, and combine them to generate distorted composite images. In this way, the entire meta network is fully trained, the Figure (2.2) show the architecture of IDeMe-Net.

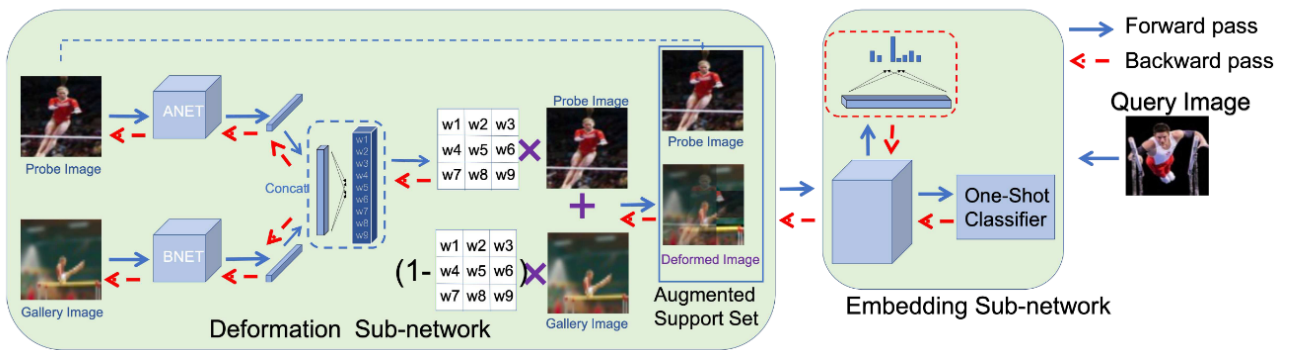


Figure 2.2: Architecture of image deformation meta-network (IDeMe-Net)[6].

As we note in table 2.2, this method showed its strength in learning from one shot, as it achieved a $59.14 \pm 0.86\%$ of accuracy, which is the highest. As for the **FSL** it achieved a good result, but not the best.

Table 2.2: Accuracy Of IDeMe-Net Model.

MiniImageNet Dataset 5- Way Accuracy (%)				
Model	Technique	1-shot	5-shots	10-shots
IDeMe-Net	Data Augmentation	59.14 ± 0.86	74.63 ± 0.74	-

2.3 Gradient-based Meta-learning Techniques

In Section(1.6), we have defined **meta-learning** a basic idea and its purpose, in this section, we explain the most important techniques gradient-based meta-learning and the results achieved. And we start with:

2.3.1 Meta-Learning LSTM (Long short-term memory)

The success of neural networks in the field of big data proves their worth, but they do not perform well in the context of a few-shot learning, to improve performance, which requires a lot of iterative steps on many examples, the meta-learner based on LSTM has been proposed [23]. (see Figure (2.3))

The reason we switched to this method is that gradient-based optimization fails:

first of which are algorithms like Adam [14], Adagrad and Adadelta. It does not perform well on updates especially when applied to non-convex optimization problems, and it doesn't have very strong guarantees of convergence speed, plus it will eventually converge to a good solution after what could be several million iterations.

Second for the network that must be randomly initialization for its parameters, which does not help convergence for good solutions after the update.

Meta-learning adjusts learning to two phases, **first** the rapid acquisition of knowledge during each task and **second** the slow information extraction across all tasks.

In order to solve gradient-based improvement problems, we need to frame this problems in a meta-learning framework.

Through this, a LSTM-based meta-learner optimizer was trained to improve the learner neural network classifier, through all the tasks, both short and long-term knowledge is captured.

To perform well during this task, the training conditions must be identical to those at the time of the test. During the meta-learning assessment, for each episode.

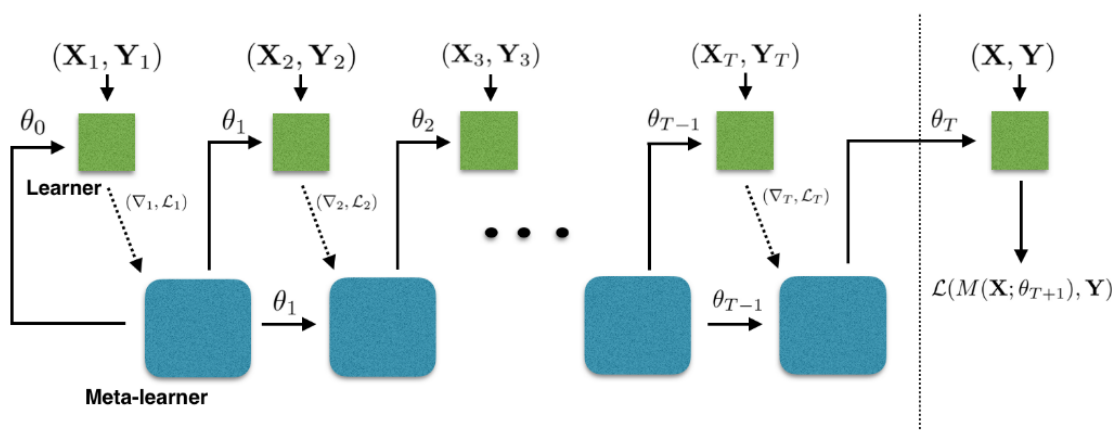


Figure 2.3: LSTM meta-learner model training [23].

The Figure (2.3) represents the front passage of the meta-learner whereas, the dashed line represents the division between the training set and the test set.

(X_i, Y_i) are the i batch of the training set while (X, Y) are all the items from the test set.

The arrow shows that there is no back-propagate during the steps when we train the meta learner.

We note in the table (2.3) that through the results achieved, it can be considered that **Meta-Learning LSTM** is competitive and respectable.

Table 2.3: Accuracy Of Meta-Learning LSTM Model.

MiniImageNet Dataset 5- Way Accuracy (%)				
Model	Technique	1-shot	5-shots	10-shots
Meta-Learning LSTM	Optimization	43.44 ± 0.77	60.60 ± 0.71	-

2.3.2 Model-Agnostic Meta-Learning (MAML)

This method trains models to quickly adapt to new tasks, and using meta-initialization, it can be useful for quickly adapting to new tasks, with just a few steps of gradient descent.

MAML does not learn an update function or learning rule, it learns the model parameters in a fully differentiable way.

Agnostic, in the sense that the method can be used in different contexts, and few shot learning is a particular case.

MAML seeks to quickly learn a new task from a small amount of new data. The idea is to train the initial parameters of the models using a different data set by giving the model parameters that have already been configured so that the model gives maximum performance when a new task comes.

The goal of MAML is to provide a good initialization of model parameters in order to achieve rapid optimization in a new task with less number of gradient steps [13].

As we can see in the Figure (2.4) parameters θ chooses the best initialization that can quickly adapt to new tasks.

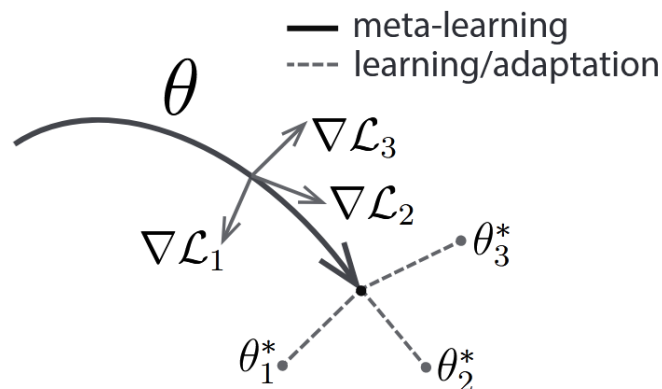


Figure 2.4: Algorithm Of Model Agnostic Meta Learning (MAML), θ represent quickly adapt to new tasks [13].

The table (2.4) shows the accuracy results of **MAML**, and as we note that there is an improvement in accuracy compared to the previous method (2.3.1), but this method is still not sufficient to provide better performance.

Table 2.4: Accuracy Of Model-Agnostic Meta-Learning (MAML).

MiniImageNet Dataset 5- Way Accuracy (%)				
Model	Technique	1-shot	5-shots	10-shots
MAML	Optimization	48.7 ± 1.84	63.00 ± 0.92	-

2.3.3 Model-Agnostic Meta-Learning ++ (MAML++)

MAML is strong, but it suffers from some problems such as instability during training, due to the sensitivity of the structures of neural networks, and also suffers from computational problems during the training period.

This is why MAML++ has been proposed, which improves computational speed during training and system stability.

MAML++ which is as flexible as automatic stability and training, it is an improvement of MAML. MAML++ greatly improves computational efficiency during training [1]. In one shot already surpasses all other methods, while additional steps allow for better performance.

Table 2.5: Accuracy Of Model-Agnostic Meta-Learning++ (MAML++).

MiniImageNet Dataset 5- Way Accuracy (%)				
Model	Technique	1-shot	5-shots	10-shots
MAML++	Optimization	$52.15+-0.26$	$68.32+-0.44$	-

The results of the table (2.5) show that the improvement over the previous method (2.3.2) was beneficial, as it achieved good accuracy over all other methods mentioned in this section (2.3).

2.3.4 Reptile

Reptile is a first-order gradient-based meta-learning algorithm. Also initializes parameters of neural network models where learning is rapid during testing and this is due to optimization of these parameters [20].

In the results obtained in this table (2.6), we note that the results are very close to MAML, this is because it only uses first-order gradient information such as first-order MAML.

Table 2.6: Accuracy Of Reptile.

MiniImageNet Dataset 5- Way Accuracy (%)				
Model	Technique	1-shot	5-shots	10-shots
Reptile	Optimization	47.07 ± 0.26	62.74 ± 0.37	-

2.3.5 Meta-SGD

Meta-SGD has the ability to learn from learning rate and update direction. Also is characterized by rapid learning and has the ability to learn effectively from examples in one step [18].

The Figure (2.5) show two-level learning process of Meta-SGD.

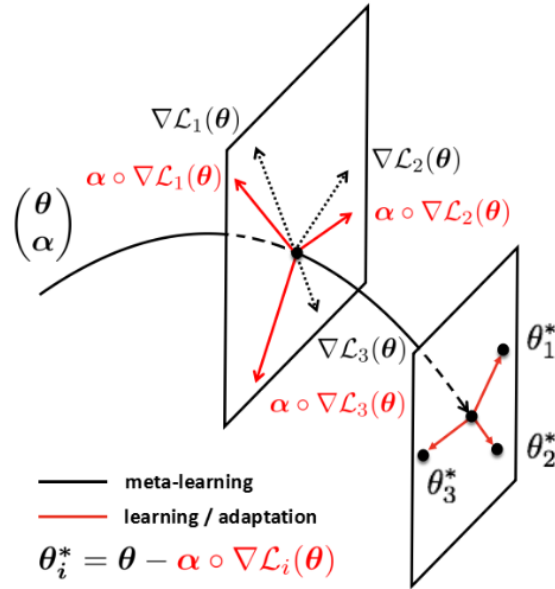


Figure 2.5: Process of Meta-SGD, meta space (θ, α) that learns the meta-learner accelerated learning is performed by the meta-learner in a learner space θ in which learners learn with a specific task [18].

The following table (2.7) shows the results obtained with Meta-SGD model, as we note, it achieved acceptable results.

Table 2.7: Accuracy Of Meta-SGD.

MiniImageNet Dataset 5- Way Accuracy (%)				
Model	Technique	1-shot	5-shots	10-shots
Meta-SGD	Optimization	50.47 ± 1.87	64.03 ± 0.94	-

2.4 Metric-Based Techniques

In this section, we present the most techniques ways to treat Metric-Based approach, these methods are mainly based on comparison or we can say learn to compare, it learns by embedding the samples to find the nearest by computing their similarity measures.

2.4.1 Siamese Neural Networks

Siamese Networks [2] are two twin networks with shared wights introduced the first time to solve the signature verification problem. Siamese networks as a one-shot image classification purpose done by [15]. The idea behind it, is to extract feature vectors from each image using two or more siamese convolutional neural network, and then, we calculate the similarity mostly with cosine similarity or euclidean distance on which loss function to train with as Contrastive loss function (2.1) or Triplet loss function.

$$\text{Contrastive loss} = (1 - Y) (D)^2 + (Y) \{\max(0, m - D)\}^2 \quad (2.1)$$

Where D is the Euclidean distance between feature vectors and m is the margin.

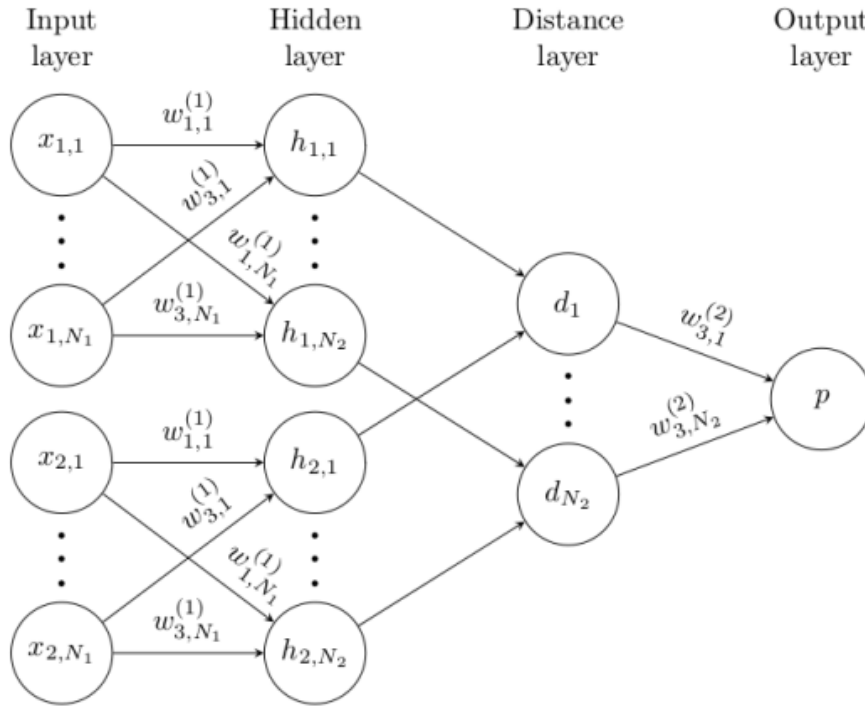


Figure 2.6: A simple Siamese Neural Networks for binary classification [15].

Table 2.8: Accuracy Of Siamese Neural Networks.

Omniglot Dataset 20- Way Accuracy (%)				
Model	Technique	1-shot	5-shots	10-shots
Siamese network	Metric-Based	92	-	-

2.4.2 Matching Networks

Matching Network [26] learns by matching the embeddings of support set and query that sequenced via a bidirectional Long-Short Term Memory (LSTM), and then we calculate the similarity between them using the Softmax classifier. (see Figure 2.7)

It's done by mapping each S where a tinny support set is defined as $S = \{x_i, y_i\}_{i=1}^k$ using c_S classifier as below (2.2):

$$c_S(\mathbf{x}) = P(y|\mathbf{x}, S) = \sum_{i=1}^k a(\mathbf{x}, \mathbf{x}_i) y_i \quad (2.2)$$

- $a(\mathbf{x}, \mathbf{x}_i)$ (2.3) is an attention mechanism, similar to non-parametric models like k-NN, the simplest form it use the Softmax through the cosine distance.

$$a(\mathbf{x}, \mathbf{x}_i) = \frac{\exp(\text{cosine}(f(\mathbf{x}), g(\mathbf{x}_i)))}{\sum_{j=1}^k \exp(\text{cosine}(f(\mathbf{x}), g(\mathbf{x}_j)))} \quad (2.3)$$

- The embedding function can be a CNN for image tasks or word embedding for NLP.

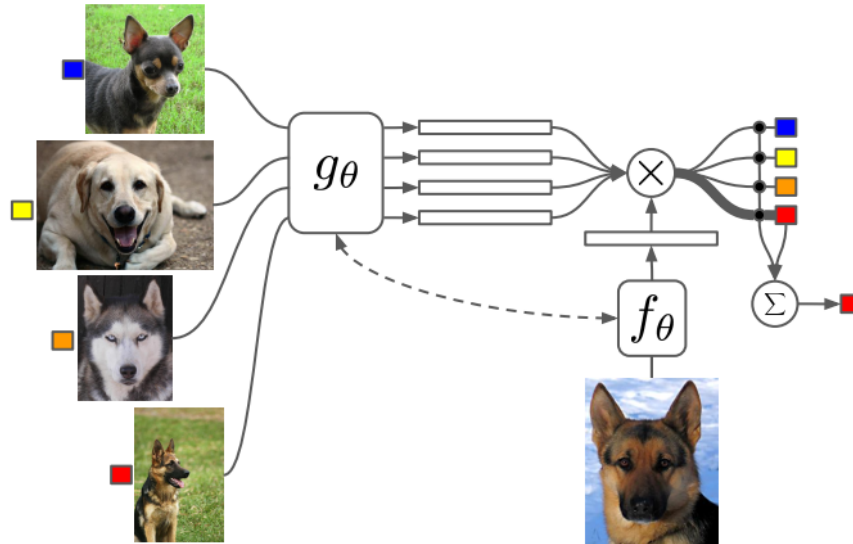


Figure 2.7: Matching Networks architecture [26].

Table 2.9: Accuracy Of Matching Network.

MiniImageNet Dataset 5- Way Accuracy (%)				
Model	Technique	1-shot	5-shots	10-shots
Matching Network	Metric-Based	43.56±0.84	55.31±0.73	-

2.4.3 Prototypical Networks

Prototypical Networks [24] it learns by calculating the prototype of each support set class $c \in \mathcal{C}$ which is the mean of embedding samples vectors, those embedding samples encoded using function f_θ , in the case of pictures, it's usually used as CNN. Then it computes the distance between the embedded query and the prototype of each class, the query belongs to the closest one. (see Figure 2.8)

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_\theta(\mathbf{x}_i) \quad (2.4)$$

Prototypical networks use the Softmax activation function, which calculates the probabilities between the distance of each class prototype over the query point \mathbf{x} :

$$P_\phi(y = c|\mathbf{x}) = \text{softmax}(-d(f_\phi(\mathbf{x}), \mathbf{c}_k)) = \frac{\exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_{k'}))} \quad (2.5)$$

Prototypical networks learn by reducing the negative log-probability:

$$\mathcal{L}(\phi) = -\log P_{\phi}(y = c|\mathbf{x}) \tag{2.6}$$

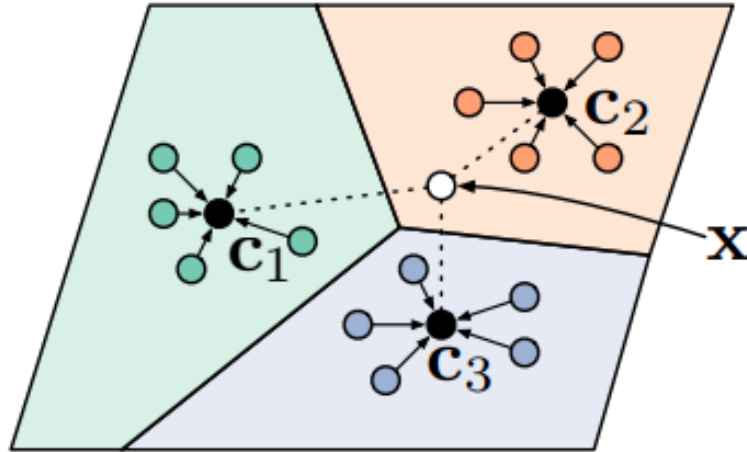


Figure 2.8: Prototypical Networks [24].

Table 2.10: Accuracy Of Prototypical Networks.

MiniImageNet Dataset 5- Way Accuracy (%)				
Model	Technique	1-shot	5-shots	10-shots
Prototypical Net	Metric-Based	49.42±0.78	68.20±0.66	-

2.4.4 Relation Network

Relation Network [25] it basically consists of two parts of embedding functions, the first $f(\phi)$ for embedding each of the the query x_j and the support set x_i samples separately, then they are grouped together as a concatenation of feature, here comes the role of the second function relation module $g(\phi)$, which gives us the similarity ratio between the query and the support set in a limited range between 0-1 called relation score. (see Figure 2.9)

Objective function: mean square error (MSE) loss used to train the model to find the real corresponding relation score $r_{i,j}$.

$$\phi, \varphi \leftarrow \operatorname{argmin}_{\phi, \varphi} \sum_{i=1}^m \sum_{j=1}^n (r_{i,j} - \mathbf{1}(y_i == y_j))^2 \tag{2.7}$$

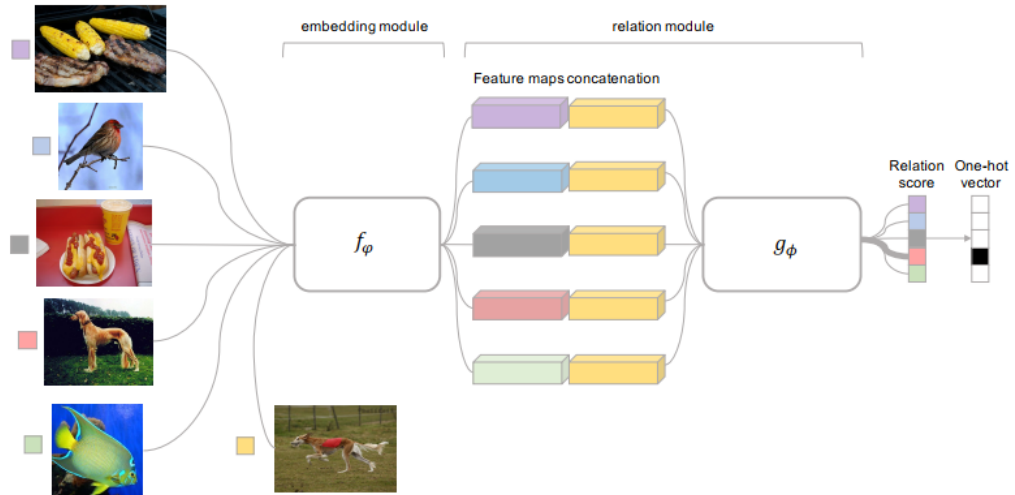


Figure 2.9: Relation Network [25].

Table 2.11: Accuracy Of Relation Network.

MiniImageNet Dataset 5- Way Accuracy (%)				
Model	Technique	1-shot	5-shots	10-shots
Relation Network	Metric-Based	50.44±0.82	65.32±0.70	-

2.5 Application

2.5.1 Computer Vision

Computer vision is one of the applications that has received great interest in the field of few-shot learning, that's why we mention many of applications related to starting from character recognition to image classification, image segmentation, texture segmentation, medical image segmentation, object detection, object recognition, image generation, image retrieval, video classification, gesture recognition, action recognition and video generation.

2.5.2 Natural Language Processing

Interest in this field has become a fertile and new field of research to be treated by few-shot learning, for example: text classification, named entity recognition, machine translation and natural language generation.

2.5.3 Robotics

In order for a robot to simulate a human, it needs to learn through a few examples, as imitation learning, visual navigation, policy learning and robot manipulation.

2.5.4 Speech Recognition

Although the applications of this field are few in FSL, but they exist such as: audio classification, text-to-speech, speech generation and speaker recognition.

2.6 Conclusion

In this chapter, we presented the most common methods in the field of FSL as well as its applications, and we also showed each method how accuracy it reached, with a focus on MiniImageNet dataset because it poses a challenge and is considered a good evaluation of the method, as the case the results achieved and the accuracy of the methods are different due to the approach taken such as Data augmentation techniques, Gradient-based Meta-learning techniques and Metric-Based techniques.

Experiment

Contents

3.1	Introduction	25
3.2	Experimentation roadmap	25
3.3	Dataset	26
3.4	Environment	27
3.4.1	Software	27
3.4.2	Hardware	27
3.5	Architecture	28
3.6	Results and Discussion	29
3.6.1	Preprocessing	29
3.6.2	Experimental results	30
3.6.3	Visualization	32
3.7	Conclusion	34

3.1 Introduction

This chapter is devoted to the purpose of presenting and discussing our experiments, we have adopted a metrics-based approach which is the **Siamese network** as a model architecture mentioned in the section (2.4.1) with the use of less data from **MNIST-FASHION** data set (Section 3.3), even studying the changes in generalization by changing some settings. Then we discuss the results obtained, the environment that we worked on, and the tools used during these experiments.

3.2 Experimentation roadmap

The goal of our experiments is to generalize through a few examples, then experiment on classes that the model did not see during the training period, we also observe and evaluate the results obtained and present them, then we verify the validity of the results.

We took the following steps to reach the goal:

1. Choosing a data set that is not widely used in the field and we have chosen **MNIST-FASHION** for this purpose.
2. Choose to download the dataset in the form of CSV, so that we can divide it according to the purpose. Another reason is that the images are small, so the model can train quickly.
3. Divide the data set into three parts in order to notice whether there is a generalization or not.
4. We reduced the size of the training data set to about 1,600 samples out of 60,000 to achieve the goal behind Few-Shot learning.
5. Concerning the training of the model, we have adopted a metrics-based approach which is the Siamese network as a model architecture with the use of Contrastive Loss Function, because it gives good results for solving generalization problems through a few examples.
6. Dividing the new dataset into pairs, so that the similar pair (those belonging to the same class) is labeled by one (1), and zero (0) in the case of a dissimilar pair (not belonging to the same class).
7. By training the model in our own way, we found that it gives good results, this is because of an initial pre-training with five attempts before starting the actual training.
8. We changed the training settings, especially the number of epochs and batch size to see its impact on the generalization.
9. Test the model to verify that there is no problem with overfitting and underfitting.
10. Testing the model on the evaluation set classes that it did not see before during training, with recording and noting the change of this result according to the epochs and batch settings.
11. We recorded some observations and presented the result of the model from predictions on the evaluation set.
12. At the end of the experiment, we came out with an important conclusion, which is the generalization is not related to the intensive training of the model.

3.3 Dataset

MNIST-FASHION [29] is a Zalando article picture data set that consists of 60,000 images for training and 10,000 for testing, all images are grayscale in size of 28x28, and each picture has one of the ten categories.

We chose to work on this data set because it is not widely used in the field, especially with Siamese Network. The data set has been downloaded through the link: <https://www.kaggle.com/datasets/zalando-research/fashionmnist> in CSV format.

We did not leave the data set as it is, but we did a **preprocessing** process of deleting two of the classes from the training set and the test set which are precisely the labels 8 represented by bag and 9 by Ankle boot, while the size of the training data set was reduced to about 200 for each class.

We also divided the data set into pairs, in order to be able to train the **Siamese Network**, the similar pair is categorized by one and the dissimilar pair is categorized by zero. Accordingly, we created a new set from that two classes called evaluation set they were cut off from the original test set (contains 2000 examples), for the purpose to testing the model until we make sure that is generalizing or not. Because the problem is not classification or measure the similarity, but generalization on classes that we have not trained on with only few examples.


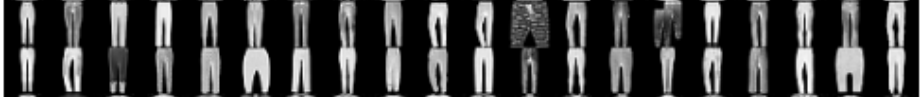








Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

Figure 3.1: MNIST-FASHION Data set [29].

3.4 Environment

3.4.1 Software

For the programming process, we used the Python programming language because it provides frameworks and libraries that facilitate the application process in the field of machine learning and deep learning, as shown below:

Python¹ As a programming language that facilitates the process of building deep learning and machine learning models.

Tensorflow² is an end-to-end open-source deep learning platform.

Keras³ is a Python-based deep learning API that runs on a high level of TensorFlow.

Pandas⁴ It is a library from Python specialized in the field of data analysis, which is also used to import and manipulating data in various formats such as CSV.

NumPy⁵ is a library from Python of mathematical functions to handle on arrays and matrices.

Matplotlib⁶ It is a Python-based library that specializes in data visualization and plotting.

Seaborn⁷ It is a Python-based library based on Matplotlib, that provides a modern visualization.

Scikit-Learn⁸ It is a Python-based library contains several machine learning algorithms that are ready to work on and we have used it in order to display the confusion matrix.

3.4.2 Hardware

To implement our experiments, we used the Google Colaboratory⁹ platform, which provides an integrated environment for deep learning, which works through the browser as Jupyter notebook and provides free GPU for facilitates training process. We used the free settings for experiments which contain the following resources:

Table 3.1: Google Colab used requirements.

CPU	Intel(R) Xeon(R) CPU @ 2.20GHz.
GPU	Tesla P100-PCIE-16GB.
DISK	78.19 GB.
RAM	12.68 GB.

We also used **Google Drive** to store the data set to facilitate the importing process by **Colab**.

¹(<https://www.python.org/>)

²(<https://www.tensorflow.org/>)

³(<https://keras.io/>)

⁴(<https://pandas.pydata.org/>)

⁵(<https://numpy.org/>)

⁶(<https://matplotlib.org/>)

⁷(<https://seaborn.pydata.org/>)

⁸(<https://scikit-learn.org/>)

⁹(<https://colab.research.google.com/>)

3.5 Architecture

We used a siamese convolutional network architecture, which is a two of identical **Convolutional neural network** shared the same weights, each CNN's creates a feature vector from an image input in our case pairs of similar images and dissimilar images to calculate the **Euclidean distance** between the two feature vectors in order to get similarity score using the sigmoid activation, for the purpose of calculating the error we used **Contrastive Loss Function** which works in a way to reduce the distance between similar inputs and increase it for dissimilar to learn to differentiate between them when new inputs come that haven't seen yet (see Figure (3.2)).

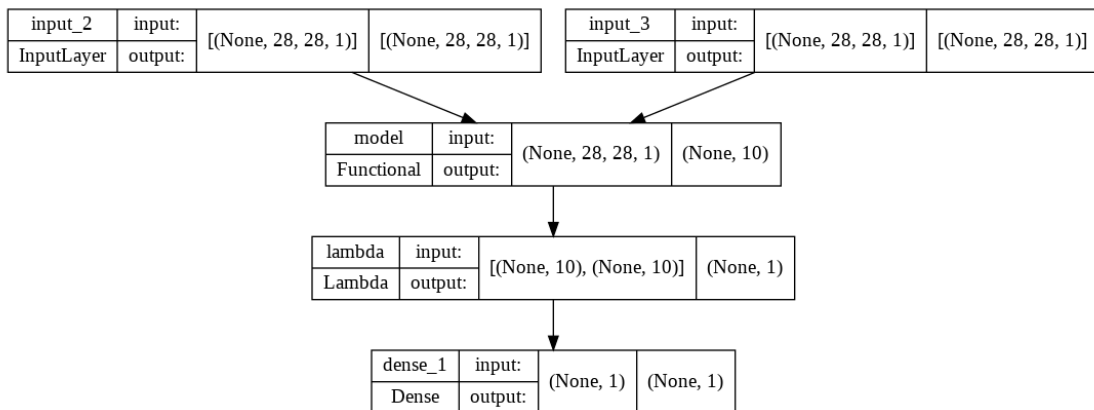


Figure 3.2: Our Siamese network architecture.

In order to get the **feature vector** we used a CNN's architecture which consists of 32 convolutional filters of 5x5 size, the next layer contain 64 of convolutional filters with 3x3 size, the two layers followed by average pooling of 2x2 size and applies ReLU as activation function, the last layer flattened into a single feature vector which represents the input, the same process is applied for both of CNN's twins Figure (3.3).

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 28, 28, 1)]	0
conv2d (Conv2D)	(None, 24, 24, 32)	832
average_pooling2d (AveragePooling2D)	(None, 12, 12, 32)	0
conv2d_1 (Conv2D)	(None, 10, 10, 64)	18496
average_pooling2d_1 (AveragePooling2D)	(None, 5, 5, 64)	0
flatten (Flatten)	(None, 1600)	0
dropout (Dropout)	(None, 1600)	0
dense (Dense)	(None, 10)	16010

 Total params: 35,338
 Trainable params: 35,338
 Non-trainable params: 0

Figure 3.3: Feature vector CNN's Encoder.

3.6 Results and Discussion

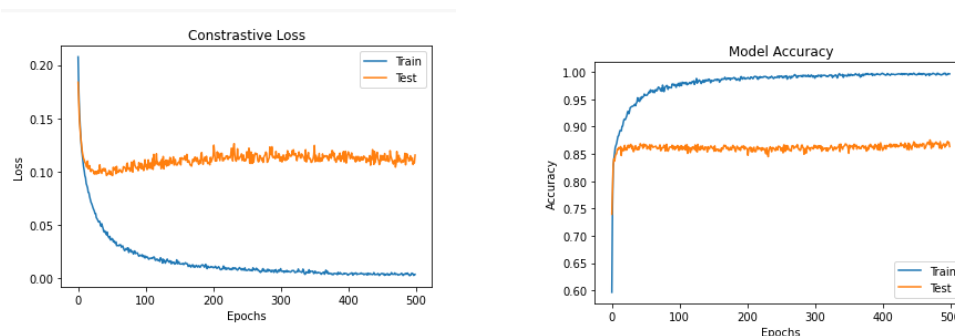
3.6.1 Preprocessing

Before we show the results obtained from our experiments, we must clarify that we have made a new division from the original data set **MNIST-FASHION**, so that it has become for us composed of three parts, the first for the train, the second for the test, and the last we called it evaluation set, to make sure that our model performs the generalization process through Few examples and do not perform the verification process on the classes seen before. This division of ours is illustrated as follows:

- **Train set** of 1600 examples labeled by 0 to 7 (**200 samples** for each class) from the original train set (contains 60000 samples) represented in the following categories:
 0. T-shirt/top.
 1. Trouser.
 2. Pullover.
 3. Dress.
 4. Coat.
 5. Sandal.
 6. Shirt.
 7. Sneaker.
- **Test set** of 8000 samples labeled from 0 to 7, it was extracted from the original test set.
- **Evaluation set** consists of 2000 examples labeled by 8 and 9 as bag. and Ankle boot, this set splitted from the original test set.

We took risks in terms of the number of test set and evaluation set samples to make sure that our model did the aim.

The aim through our experiments, is to study the effect of number of epochs and batch size on generalization through the evaluation set. We chose the epoch 100, 50 and 20 as based on our experiments, because we have tried the epoch 500, where we noticed the stability of the accuracy at 100 as shown in the following illustrations plots (Figure 3.4).



(a) The model's loss plot.

(b) The model's accuracy plot.

Figure 3.4: The training and testing accuracy and loss plots using 10 batch.

3.6.2 Experimental results

We did our experiments on the Siemese network using the following settings:

- Data set: **MNIST-FASHION** in the Section (2.4.1) and (3.6.1).
- Feature vector: Twin of Convolutional Neural Network see the section (3.5).
- Loss function: **Contrastive Loss Function** in (2.1).
- Distance: **Euclidian distance** (1.8).
- Samples: 1600 samples, 200 of each class, 3200 pairs.
- Optimizer: Adam with default parameters.
- batch size: 1, 5, 10, 20, 30.
- number of epochs: 20, 50, 100.
- Environment: for software see (3.4.1) and hardware (3.4.2).

Table 3.2: The accuracy of the training and testing and evaluation set.

epochs	batch	Training	Test	Evaluation
20	1	92.78%	86.83%	81.25%
20	5	90.72%	86.18%	80.00%
20	10	88.78%	85.03%	83.57%
20	20	90.56%	86.96%	83.17%
20	30	86.87%	84.82%	77.55%
50	1	95.69%	85.51%	74.60%
50	5	91.84%	83.12%	80.05%
50	10	91.44%	84.60%	77.30%
50	20	92.28%	83.12%	78.40%
50	30	92.81%	86.41%	79.68%
100	1	98.34%	87.58%	79.50%
100	5	97.22%	85.72%	79.18%
100	10	98.44%	87.00%	75.00%
100	20	97.06%	87.97%	82.60%
100	30	97.75%	87.73%	78.97%

- Through the results obtained by our model, we can say that we have obtained very well results that have achieved the goal of the experiments, which is to generalize through a few examples.
- We also reached a training method that gives good results, which is to pre-train the model in a few instances before starting the actual training.
- As we noticed after training the model on several settings, it shows us that test accuracy settles at a certain percentage that does not exceed 87%, although training may exceed this number when we increase the number of epochs.

- The most important thing we noticed through our experiments is that the accuracy on the evaluation set is not related to intensive training, but through a little training, we got good results, close to 84% in the case of 20 epochs with 10 batch, while in the case of 100 epochs we got 75.00%.



Figure 3.5: The graph for the performance over 20 epochs.

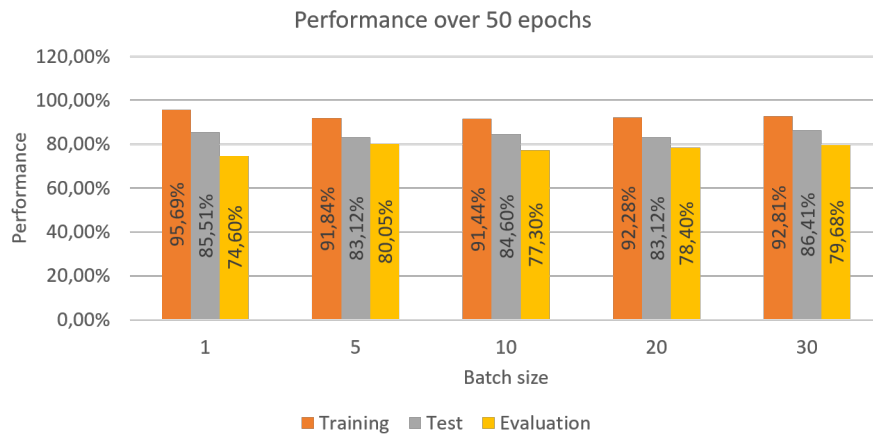


Figure 3.6: The graph for the performance over 50 epochs.

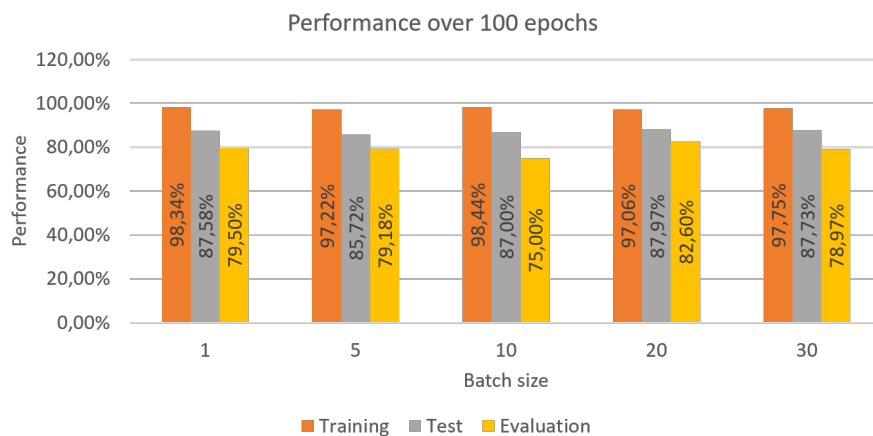


Figure 3.7: The graph for the performance over 100 epochs.

3.6.3 Visualization

In this section, we present our results, but in an illustrated way that is easy to extrapolate. These results were recorded through our model with the following settings batch size of 10 and 20 epochs, which recorded the best possible accuracy especially on the evaluation set. We start by presenting our results in the form of a **Confusion Matrix** which briefly demonstrates the accuracy of our model In the various divisions of our data set, as described below:

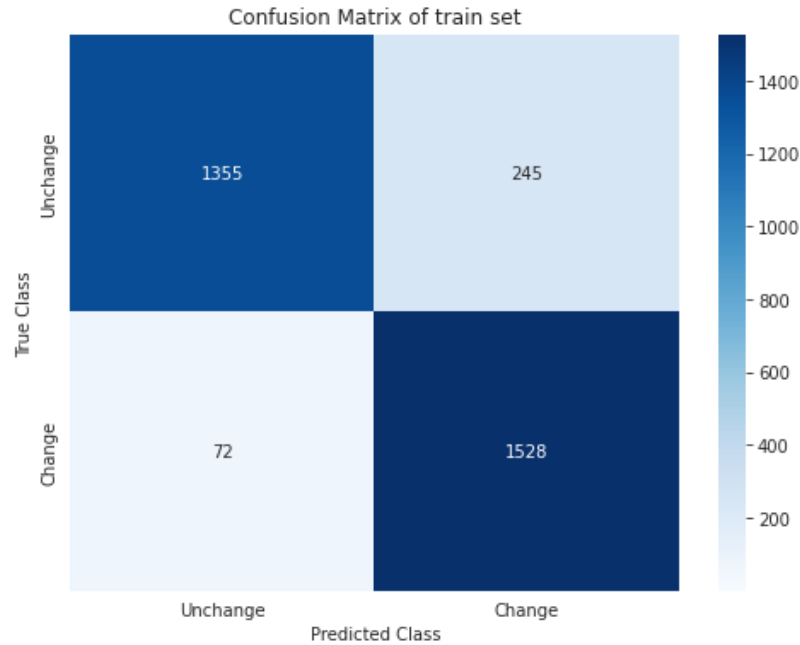


Figure 3.8: The confusion matrix for the settings of 10 batch size and 20 epochs performance on the train set.

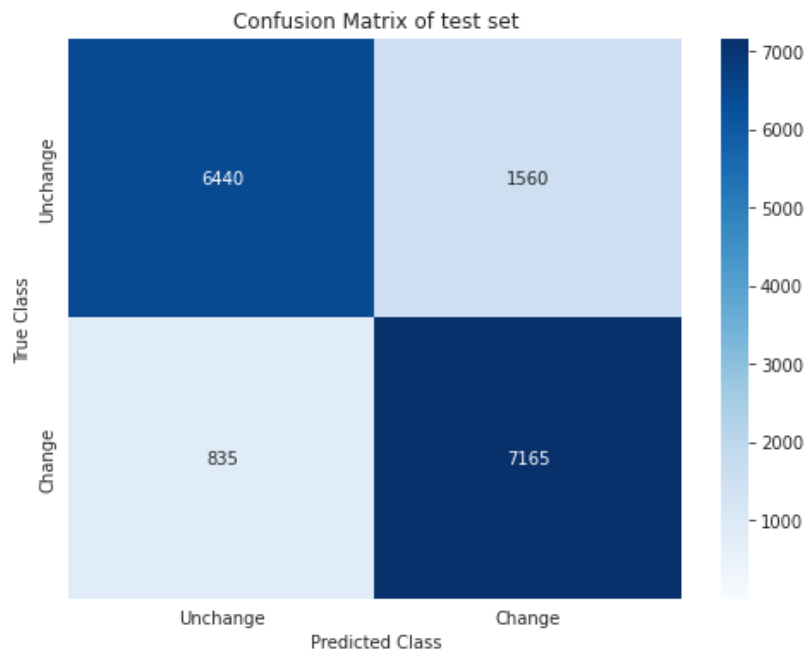


Figure 3.9: The confusion matrix for the settings of 10 batch size and 20 epochs performance on the test set.

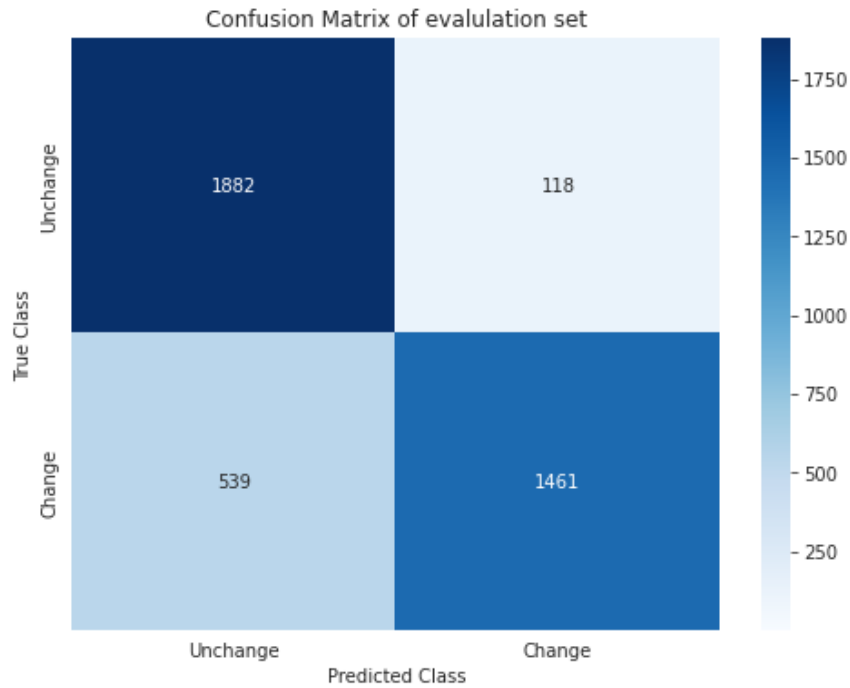


Figure 3.10: The confusion matrix for the settings of 10 batch size and 20 epochs performance on the evaluation set.

In this **visualization** see the Figure below(3.11), we show the final result of our model, which is to differentiate between classes that were not seen before, through training from few examples. When the similarity ratio approaches one, they are similar, and less than 0.5 means there is a dissimilarity between them.

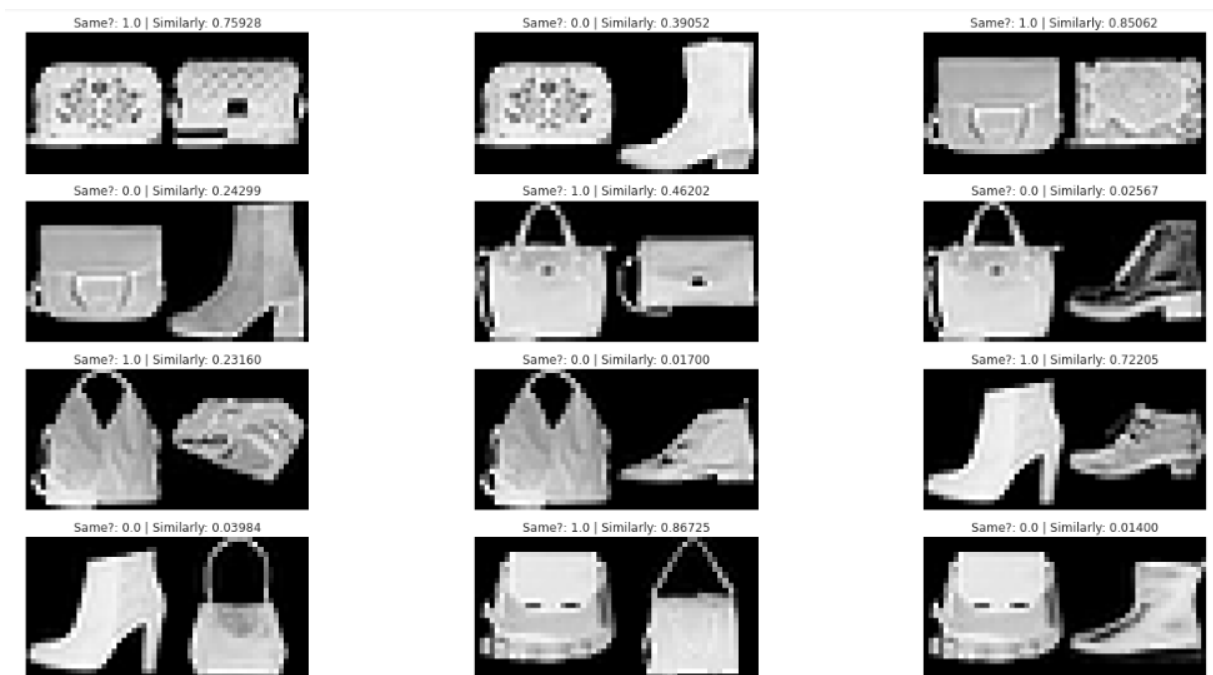


Figure 3.11: Visualization of differentiate between classes from evaluation set.

3.7 Conclusion

In this chapter, we did experiments on Siamese Networks using Contrastive Loss to make sure that it gives general results on classes that we have not trained on before, and this is through the data set that does not contain many examples such as Fashion MNIST, we also divided the data set in our own way and we reduced the size of the training data set to about 200 examples for each class to enable us to observe the results.

Finally, we made some observations through our experiments and also we display some visualization.

Conclusion

We have studied in this work an important topic, which is learning through a small amount of data or as it is called Few-shot learning, where we have tried to provide a comprehensive view of the field and the beginning was to define the concept of FSL and what are k-shot n-way, support set and query set, without forgetting its challenges, we have also outlined the differences between the most important variations, we have even gone into the depths by abstracting the most important approaches in the field such as Meta learning and Metric learning, up to transfer learning which is also included, we have presented the most important data sets in FSL.

As for related work, we have mentioned in detail the methods used in the field, not all but the main ones, with a specification of the result of each of them, and we have finished the theoretical side by mentioning some important applications.

Our experiments focused on studying the generalization problem through a small number of training data using the Siamese neural network, especially concerning the MNIST Fashion dataset, where we noticed that it was not widely used in the field, so we divided it in our way and reduced the amount of data from 6000 samples for each class down to 200, as well as we studied the model based on two variables: the number of epochs and the batch size. At the end of our experiments, we obtained the result that shows that generalization is not related to intensive training, because with a small training of 20 epochs and 10 batches, we obtained an accuracy of approximately 84%.

References

- [1] Antreas Antoniou, Harrison Edwards, and Amos Storkey. “How to train your MAML”. In: *arXiv preprint arXiv:1810.09502* (2018).
- [2] Jane Bromley et al. “Signature verification using a” siamese” time delay neural network”. In: *Advances in neural information processing systems* 6 (1993).
- [3] Shuvam Chakraborty et al. “Efficient conditional pre-training for transfer learning”. In: *arXiv preprint arXiv:2011.10231* (2020).
- [4] Wei-Yu Chen et al. “A Closer Look at Few-shot Classification”. In: *ArXiv abs/1904.04232* (2019).
- [5] Yutian Chen et al. “Learning to learn without gradient descent by gradient descent”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 748–756.
- [6] Zitian Chen et al. “Image deformation meta-networks for one-shot learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8680–8689.
- [7] Mehdi Cherti and Jenia Jitsev. “Effect of Pre-Training Scale on Intra-and Inter-Domain Full and Few-Shot Transfer Learning for Natural and Medical X-Ray Chest Images”. In: *arXiv preprint arXiv:2106.00116* (2021).
- [8] Guneet S Dhillon et al. “A baseline for few-shot image classification”. In: *arXiv preprint arXiv:1909.02729* (2019).
- [9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-agnostic meta-learning for fast adaptation of deep networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 1126–1135.
- [10] Bharath Hariharan and Ross Girshick. “Low-shot visual recognition by shrinking and hallucinating features”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 3018–3027.
- [11] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [12] Mike Huisman, Jan N Van Rijn, and Aske Plaat. “A survey of deep meta-learning”. In: *Artificial Intelligence Review* 54.6 (2021), pp. 4483–4541.
- [13] Shruti Jadon. “An overview of deep learning architectures in few-shot learning domain”. In: *arXiv preprint arXiv:2008.06365* (2020).
- [14] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *CoRR abs/1412.6980* (2015).

- [15] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. “Siamese neural networks for one-shot image recognition”. In: *ICML deep learning workshop*. Vol. 2. Lille. 2015, p. 0.
- [16] Xiaomeng Li et al. “Revisiting Metric Learning for Few-Shot Image Classification”. In: *Neurocomputing* 406 (2020), pp. 49–58.
- [17] Xiaoxu Li et al. “Deep Metric Learning for Few-Shot Image Classification: A Selective Review”. In: *ArXiv abs/2105.08149* (2021).
- [18] Zhenguo Li et al. “Meta-SGD: Learning to Learn Quickly for Few Shot Learning”. In: *ArXiv abs/1707.09835* (2017).
- [19] Jiang Lu et al. “Learning from Very Few Samples: A Survey”. In: *ArXiv abs/2009.02653* (2020).
- [20] Alex Nichol, Joshua Achiam, and John Schulman. “On First-Order Meta-Learning Algorithms”. In: *ArXiv abs/1803.02999* (2018).
- [21] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. “Tadam: Task dependent adaptive metric for improved few-shot learning”. In: *Advances in neural information processing systems* 31 (2018).
- [22] Archit Parnami and Minwoo Lee. “Learning from Few Examples: A Summary of Approaches to Few-Shot Learning”. In: *ArXiv abs/2203.04291* (2022).
- [23] Sachin Ravi and H. Larochelle. “Optimization as a Model for Few-Shot Learning”. In: *ICLR*. 2017.
- [24] Jake Snell, Kevin Swersky, and Richard Zemel. “Prototypical networks for few-shot learning”. In: *Advances in neural information processing systems* 30 (2017).
- [25] Flood Sung et al. “Learning to Compare: Relation Network for Few-Shot Learning”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 1199–1208.
- [26] Oriol Vinyals et al. “Matching networks for one shot learning”. In: *Advances in neural information processing systems* 29 (2016).
- [27] Catherine Wah et al. “The caltech-ucsd birds-200-2011 dataset”. In: (2011).
- [28] Yaqing Wang et al. “Generalizing from a few examples: A survey on few-shot learning”. In: *ACM computing surveys (csur)* 53.3 (2020), pp. 1–34.
- [29] Han Xiao, Kashif Rasul, and Roland Vollgraf. “Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms”. In: *ArXiv abs/1708.07747* (2017).
- [30] Han-Jia Ye et al. “Few-Shot Learning with a Strong Teacher”. In: *IEEE transactions on pattern analysis and machine intelligence* PP (2022).
- [31] Zhongqi Yue et al. “Interventional few-shot learning”. In: *Advances in neural information processing systems* 33 (2020), pp. 2734–2746.